# Dynamic Penalty Based GA for Inducing Fuzzy Inference Systems

Tomás Arredondo V.[1], Félix Vásquez M.[2], Diego Candel C.[2], Lioubov Dombrovskaia[2], Loreine Agulló[3], Macarena Córdova H.[3], Valeria Latorre-Reyes[3,4], Felipe Calderón B.[2], and Michael Seeger P.[3]

[1] Departamento de Electrónica
e-mail: tarredondo@elo.utfsm.cl
[2] Departamento de Informática,
[3] Millennium Nucleus EMBA, Departamento de Química
Universidad Técnica Federico Santa María
Av. España 1680, Valparaíso, Chile
[4] Universidad de Magallanes, Punta Arenas, Chile

**Abstract.** Fuzzy based models have been used in many areas of research. One issue with these models is that rule bases have the potential for indiscriminant growth. Inference systems with large number of rules can be overspecified, have model comprehension issues and suffer from bad performance. In this research we investigate the use of a genetic algorithm towards the generation of a fuzzy inference system (FIS). We propose using a GA with a dynamic penalty function to manage the rule size of the fuzzy inference system (FIS) while maintaining the exploration of good rules. We apply this method towards the generation of a fuzzy classifier for the search of metabolic pathways. The GA based FIS includes novel mutation and a penalty based fitness scheme which enables the generation of an efficient and compact set of fuzzy rules. Encouraging implementation results are presented for this method as compared with other classification methods. This method should be applicable to a variety of other modelling and classification problems.

**Keywords:** Fuzzy logic, inference system, genetic algorithm, system modelling.

## 1 Introduction

A common task in bioinformatics research consists in the search and identification of genes encoding the enzymes of metabolic pathways of interest in recently sequenced genomes. Towards this purpose there exists a diverse set of tools, databases (e.g. KEGG, NCBI) and applications (e.g. BLAST, Artemis, Vector NTI). The integration of these resources is a current area of interest given the complexity and skill required to manually utilize all these means efficiently. One of these initiatives is GeXpert [1, 2], an implementation of an integration framework that involves a systematic search scheme for the identification of genes encoding enzymes of metabolic pathways.

One of the tools that GeXpert includes is a gene evaluator based on a set of fuzzy rules [3] that attempts to estimate how good a candidate DNA coding sequence is for a given enzyme in a metabolic path. Until now, the rules used by this classifier were created manually from the knowledge provided by a group of bioinformatics researchers. Our current proposal is to use a penalty based GA to data-mine the classification rules from the set of previously analyzed genes for the organism under investigation. To the best of our knowledge, a GA based agent for fuzzy inference system (FIS) rule data-mining and training in bioinformatics is a novel application that has not been attempted before. We have also introduced new mutation and fitness schemes in this GA which attempt to emulate how a researcher would operate. Finally we have introduced a dynamic penalty function in order to constrain the number of fuzzy rules generated without affecting the initial exploration of the system. Initial results have shown that this algorithm is capable of synthesizing an efficient and compact fuzzy rule set when compared to other methods.

In section 2, we briefly explain the integration architecture and the implementation of GeXpert. Section 3 describes the FIS used in GeXpert. Section 4 describes the fuzzy trainer module that was implemented. In section 5, we describe the experiments performed to validate our work. In section 6, we explain our test results. Finally, in section 7 some conclusions are drawn and future research directions are presented.

## 2 Integration Framework Architecture

Towards metabolic reconstruction based on sequenced genomes, GeXpert implements an efficient research methodology as has been proposed in [4]. This methodology considers the following integrated workflow:

1. Create or import (e.g. KEGG) the metabolic pathway of interest in the organism under study.
2. Download the sequence of the enzyme (or subunits) being searched from a database (e.g. GenBank).
3. Perform alignments using tblastn between the selected proteins and the genome under investigation.
4. Using the fuzzy classifier, classify the sequences found according to their alignment scores: Identity, E-value, Gaps, Bit-Score.
5. Verify (e.g. using ARTEMIS) if there is an coding sequence (CDS) containing the sequence of the best alignments that were found.
6. If a CDS is found, the sequence is tested with a blastp alignment versus the proteins in a public database (e.g. GenBank).
7. If the proteins found that are aligned with the CDS sequence correspond with the enzyme that was searched for initially and there are documentation references (e.g. PUBMED) that validate its existence as a non putative protein then this enzyme or subunit is considered to have been found in the organism.

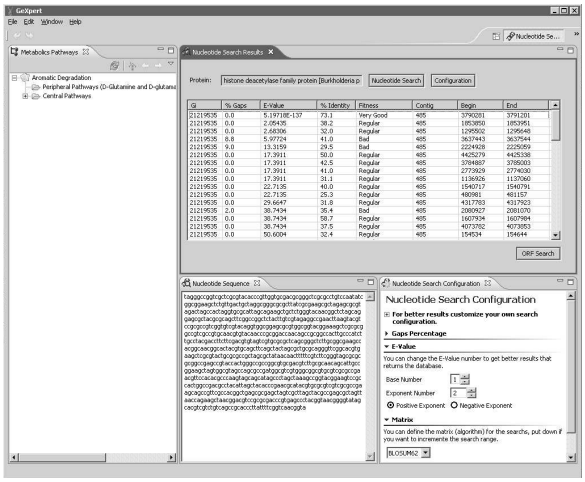


**Fig. 1.** GeXpert FIS graphical interface

The integration architecture includes three layers: the presentation layer, the logical layer and the data layer [2]. In our implementation of the presentation layer several graphical interfaces (e.g. as seen in Fig. 1) have been developed (e.g. to edit metabolic pathways, protein searches, visualizing CDS results and blast alignments). The logical layer contains the business logic relating with the different objects that encapsulate the applications and utilities used in the research process (e.g. relating with alignments, fuzzy classification, and CDS searches). The data layer manages all the interfaces (e.g. App Call, JDBC, SOAP) with various internal and external data sources (e.g. KEGG, GenBank, NCBI) and applications (e.g. BLAST, CN3D, Artemis). In this implementation, the training agent initiates interfaces with the core engine which is where the fuzzy system resides.

## 3 Fuzzy Inference System Workflow

The fuzzy inference system workflow is shown in Fig. 2. After a protein that forms part of a metabolic pathway that is being reconstructed has been identified, the researcher will perform a GenBank search using tblastn (Fig. 2(a)). This is to verify that the gene encoding the protein in question is found in another organism. Next, a list of possible candidates (each including four BLAST output values) will be transmitted to the fuzzy engine (Fig. 2(b)). The fuzzy engine will analyze the parameters and will make a recommendation to the user with respect to the quality of each of the candidates (Fig. 2(c)). Once the expert concludes (True of False) whether the gene encodes or not an enzyme or subunit of the metabolic pathway then the parameters, fuzzy recommendation and the expert conclusion are stored in the database as a training case (Fig. 2(d)).

If sufficient database entries are available, the scheduling agent or a user can initiate the fuzzy engine training process with the goal of tuning the recommendations generated by the fuzzy inference engine (Fig. 2(e)-2(g)). This tuning can be periodically performed as the size of the training set increases.

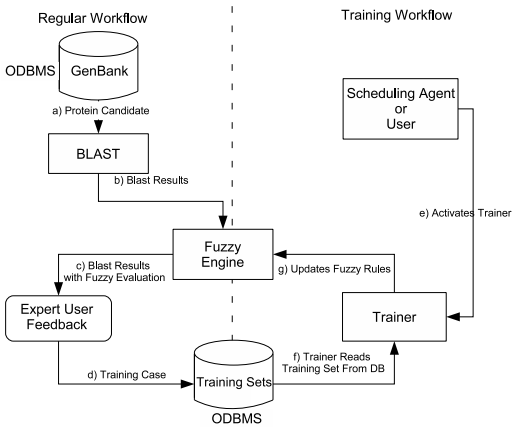


**Fig. 2.** FIS workflow

## 4 Fuzzy Trainer Module Design

Fuzzy inference systems (FIS) are generally composed of: fuzzy rules, membership functions and a form of fuzzyfication/deffuzyfication. Any of these elements could be updated or modified towards tuning the FIS. In [5] the simultaneous modification of the membership functions and the rule base is proposed to optimize an FIS. The approach taken in our work follows [6, 7] in which only the rule base is actualized during training.

### 4.1 Fuzzy Engine Structure

**Fuzzy Rules.** The rule base of the system follows the Mamdani method and has the following structure:

**IF** *E-value* is INVALUE **AND** *Bit-Score* is INVALUE **AND** *Identity* is INVALUE **AND** *Gaps* is INVALUE **THEN** *Output* is OUTVALUE.

Where INVALUE is a membership values with one of the following possible values {Very Low, Low, Medium, High, Very High}. OUTVALUE is the expected result for said combination of fuzzy inputs. OUTVALUE can take the following values {Very Bad, Bad, Regular, Good, Very Good}.

**Membership Functions.** Fuzzy membership functions take input parameters from $[0, 1]$. Input parameters that have different ranges are normalized before they are introduced into the FIS. Membership functions are triangular as is seen in Fig.3. The defuzzifier uses the centroid method using the maximum of all activated functions.

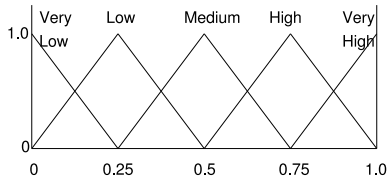


**Fig. 3.** Membership functions

### 4.2 GA Trainer Structure.

As mentioned previously, the FIS is trained using a genetic algorithm. In this algorithm, each individual is composed of $n$ genes, where each gene corresponds to a fuzzy rule.

**GA Individuals.** In our encoding, each fuzzy rule is defined by a set of 5 values that identify the different fuzzy sets for each of the four BLAST output values (E: E-value, B: Bit Score, I: Identity, G: Gaps) and the result (O: Output). These four output values are scored according to the alignment between sequences. Their relationship to whether a CDS truly encodes a given enzyme is not trivial to determine. As seen by the example in Fig.4, the fuzzy values {Very Low, ..., Very High} and {Very Bad, ..., Very Good} are represented by {1, 2, 3, 4, 5} respectively.

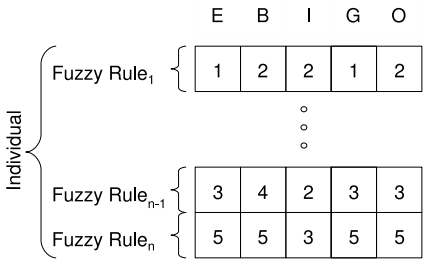


**Fig. 4.** GA codification

**Genetic Operators.** In our scheme, we have used one crossover operator and three different mutation operators. The mutation operators used generate a variable number of rules used which is essential to explore the problem search space given that we are using fixed membership functions. Using fixed membership functions and a variable number of rules is in our opinion more intuitive than other methods and better emulates how a human expert would update the FIS.

*Crossover.* In our system we are using a simple one point crossover that selects two individuals, $\text{ind}_1$ and $\text{ind}_2$, from the current population. Randomly selecting a value $e$, that is found in the interval $[0, a]$, with $a = \min(\text{size}(\text{ind}_1), \text{size}(\text{ind}_2))$. After this, the first $e$ number of rules are exchanged between the individuals.

*Mutation.* The three mutation operators implemented are the following:

1. Adding Rules. This operator increases the rules of an individual by adding new random rules (genes) up to a specified factor $\mu$ of the original number of rules.
2. Removing Rules. This operator reduces the number of rules of an individual by removing up to a certain percentage factor $\nu$ of number of rules (genes). The rules are selected at random for removal.
3. Changing rules. This operator selects a random number of genes for modification in each individual. This operator selects the first $b$ rules of an individual and mutates them, where $b$ is a whole number chosen at random in the interval $[0, \text{size}(\text{ind})]$. The mutation consists in choosing and changing at random one of the parameters that compose it.

The values for $\mu$ and $\nu$ were empirically chosen to be 0.33 as to not be too destructive but at the same time to allow for significant changes in the individuals.

**Fitness Function.** The fitness function used consists of two parts, the first is the sum of the hits ($sh$) obtained using the FIS. The value of $sh$ is calculated based on the training cases $tc_i$ that are stored in the database where $tc = [tc_1, tc_2, \ldots, tc_n]^T$ are generated as shown in Fig. 2. The second part consists of a penalty value that is applied to individuals which have a large number of rules ($rp$). To obtain the value of $sh$ we sum the points obtained by the individual when we apply the values from each training example $tc_i$. The value of $sh(tc_i, O_i)$ is calculated using the values from scoring Table 1, where $tc_i$ will have the values True or False and where $O_i$ is the output expected for the $i$-th training example. Hence, $sh(tc)$ is calculated according to

$$sh(tc) = \sum_{i=1}^{n} sh(tc_i, O_i). \tag{1}$$

A bivariate sigmoid function is used for penalizing individuals with an excess number of rules depending on the current iteration number of the GA. This equation is defined as

$$rp(x, y) = \left( \frac{\alpha(y - \beta)}{1 + e^{(x-\gamma)\delta}} \right), \tag{2}$$

where $x$ indicates the number of rules that compose the individual, $y$ is the

| | Very Bad | Bad | Medium | Good | Very Good | No Rule Fired |
|---|---|---|---|---|---|---|
| True | 3 | 4 | 5 | 6 | 7 | 0 |
| False | 7 | 6 | 5 | 4 | 3 | 0 |

**Table 1.** Scoring table for truth value assignation

iteration number and $\alpha, \beta, \gamma, \delta$ are parameters. The parameter $\beta$ indicates up to which iteration exploration should be favored (thus rewarding new rule creation and exploration). Past this iteration, the penalty for the number or rules begins to be effective. The other parameters are for sigmoid scaling. Finally, the fitness of an individual can be expressed as

$$F = sh(tc) - rp(x, y) \tag{3}$$

The parameters used in our experiments were empirically determined. Six tests were performed with various GA parameter values (e.g. crossover and mutation probability) and in all situations only marginal fitness differences were seen (e.g. $1 - 5\%$). The number of elite individuals was chosen as to not cause too much premature convergence. In the case of $\alpha$ (penalty scaling) and $\beta$ (beginning iteration for penalization) another six combinations were investigated. All resulted in minor differences in final results (e.g. considering true positives and negatives). Finally these were set to: $\alpha = 0.08, \beta = 80, \gamma = 100, \delta = -0.06$. $\gamma$ and $\delta$ were set such that the penalty curve have an impact within a range of $0 - 225$ iterations.

**GA parameters.** The GA parameters were set as follows:

- Crossover probability=0.6.
- Mutation probability=0.26.
- Maximum generations=225.
- Population size=30.
- Number of elite individuals=4.
- Roulette selection method.

## 5 Experiments

Two experiments were performed in order to test our approach. The first was to compare the efficiency of using the current GeXpert system versus the previous manual research method. Results considered compared timings for 6 different metabolic pathways which genes are were organized in operon(2), distributed (2) and non existent(2). These results were taken from seven users of varying competence. Three pathways were researched using the manual method and the other three were researched using GeXpert.

The other experiment was done in order to examine the training system developed. For this experiment, we used 248 sample genes. Out of these, 124 genes (true positives or TP) are encoding proteins of metabolic pathways of the bacteria *Burkholderia xenovorans* strain LB400 [8]. The other 124 genes were determined not be part of its genome (true negatives or TN). Out of this sample space, the system was trained with 62 true positive (TP) genes chosen at random and 62 true negatives that were also chosen at random. The other 124 genes (50% TN and 50% TP) were used as test cases to validate the method.

## 6 Results Obtained

In this section we present the results for the two experiments previously described. We present pathway search times and compare classifier performance.

The efficiency experiment showed that for our six researchers using GeXpert provides an average 400% faster search time than when using the previous manual research method. This vast improvement in time was tempered by a small average increase in enzyme error recognition (about 15%) which could be explained due to a lack of system familiarity or the reduced amount of spare time (e.g. idle time waiting) when using GeXpert. This error is reduced or even non existent in those researchers who were most familiar with GeXpert.

In the fuzzy rule performance experiment, we compare five different classifiers: two sets of fuzzy rules (GA-1 and GA-2) that were generated by the GA training method, two human generated rules (H-Naive and H-Expert) and a standard SVM classifier implementation (Weka SMO [9]). GA-1 uses the method described previously but the fitness value ($F$) does not include the penalty function ($rp$) while GA-2 does. The H-Naive rules were constructed by analyzing in a short timespan (about 3 hours) the training data with the aim of covering the entire spectrum of possibilities. For the H-Expert rules a more detailed analysis was performed (about 6 hours) of the training data. For this analysis, fuzzy rules were created based on the input given by expert biochemists that work in the field of gene searching. Table 2 shows the results that were obtained including true positives (TP), true negatives (TN), false positives (FP), false negatives (FN) as well as the total percentage true values obtained. For each test case of GA-1 and GA-2 an average of 40 test experiments are shown.

| Classifier | # Rules | SD Rules | TP% | FN% | TN% | FP% | Total True% | Total False% |
|---|---|---|---|---|---|---|---|---|
| SVM | NA | NA | 86.0 | 14.0 | 100.0 | 0 | 92.7 | 7.3 |
| H-Naive | 400 | 0 | 86.0 | 14.0 | 30.0 | 70.0 | 58.0 | 42.0 |
| H-Expert | 163 | 0 | 70.0 | 30.0 | 100.0 | 0.0 | 85.0 | 15.0 |
| GA-1 | 93 | 31 | 89.8 | 10.2 | 98.5 | 1.5 | 94.1 | 5.9 |
| GA-2 | 60 | 15 | 89.9 | 10.1 | 98.9 | 1.1 | 94.4 | 5.6 |

**Table 2.** FIS result comparison

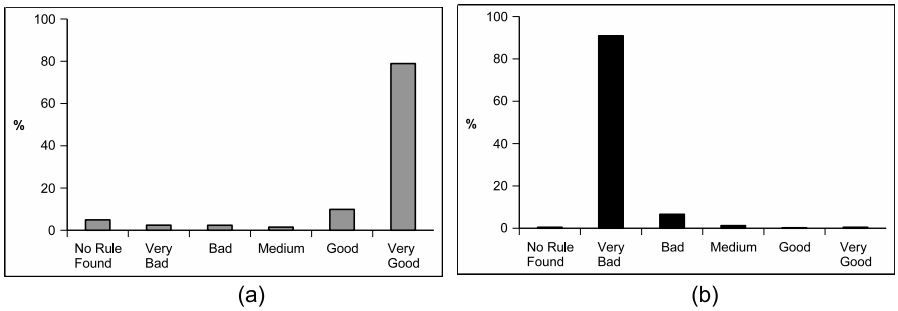


**Fig. 5.** Results obtained for the best set of rules (GA-2). In (a) we show true positives (Good, Very Good) and false negatives (Very Bad, Bad, Medium). In (b) we show true negatives (Very Bad, Bad, Medium) and false positives (Good, Very Good).

## 7 Conclusions

As seen by the test results, using the integrated search environment provides for much faster metabolic pathway search times than previous manual methods without much penalty in terms of additional error. The small error introduced was found in users who had less experience with the application and should be reduced as they gain more experience with it.

From these initial results it can be observed that using our training best algorithm (GA-2) we have obtained an overall improvement of 9% over the results obtained from the human expert rule sets. When compared with the other classifiers, GA-2 also provided good results with the lowest FN and FP values. GA-2 and its included penalty also produced smaller rule sets that had a lower standard deviation with respect to the number of rules generated (e.g. more predictable and smaller execution times). Otherwise results between GA-1 and GA-2 generated rule sets seem equivalent. GA-2 has a somewhat better overall performance than SVM given that its results are comparable when classifying true negatives but are better in the classification of true positives.

In our observations, in the case of GA-2 the number of rules is significantly lower than the rules that would be generated by other methods. The resulting reduction in FP and FN in GA-2 could be due to a reduction in the overspecification of the model regarding the training data or due to a reduction in rule contradictions. Other GA based fuzzy optimization methods [6, 7] (without such a penalty function) could possibly benefit from such an approach. Also a reduction in the number of rules makes the FIS much more intuitive and easier to understand. Another benefit of the system is the flexibility provided by having multiple sets of parameter values (e.g. scoring table values, $F$ function parameters) in order to penalize FP and FN in a differentiated manner.

Reducing rule bloat is an objective that seems to have been accomplished without any real damage to classification capability. In the future, we will focus on implementing our penalty based method on other fuzzy based classification

applications (e.g. robotics [10]) to validate whether this approach is applicable to a variety of other problems. In general the system was robust to parameter variations which leads us to believe that they should work in a variety of situations. Function penalty is calculated in each generation and for each individual during training, but being a simple evaluation should not greatly increase the total execution time. Also, run time savings obtained through rule reductions should more than compensate for the training time increase.

## Acknowledgements

This research was partially funded by the DGIP of UTFSM (230726 and 240726) and by Fundación Andes, Milenio P04/007-F (Mideplan).

## References

1. GeXpert: Gexpert project website. (`http://sourceforge.net/projects/gexpert`)
2. Arredondo, T., Seeger, M., Dombrovskaia, L., Avarias, J., Calderón, F., Candel, D., Muñoz, F., Latorre, V., Agulló, L., Cordova, M., Gomez, L.: Bioinformatics integration framework for metabolic pathway data-mining. Lecture Notes in Artificial Intelligence **4031** (2006) 917–926
3. Zadeh, L.: Fuzzy sets. Information and Control **8** (1965) 338–353
4. Sun, J., Zeng, A.P.: Identics - identification of coding sequence and *in silico* reconstruction of the metabolic network directly from unannotated low-coverage bacterial genome sequence. BMC Bioinformatics **5** (2004) 112
5. Peña-Reyes, C.A., Sipper, M.: Designing breast cancer diagnostic systems via a hybrid fuzzy-genetic methodology. In: Proceedings of 1999 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'99). Volume I., IEEE Press, Piscataway, NJ (1999) 135–139
6. Herrera, F., Lozano, M., Verdegay, J.L.: Generating fuzzy rules from examples using genetic algorithms. Proc. IPMU'94 (5th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems) (1994) 675–680
7. Wang, W.J., Yen, T.G., Sun, C.H.: A method of self-generating fuzzy rule base via genetic algorithm. Control Conference, 2004. 5th Asian **3** (2004) 1608–1615
8. Chain, P.S.G., Denef, V.J., Konstantinidis, K.T., Vergez, L.M., Agulló, L., Reyes, V.L., Hauser, L., Córdova, M., Gómez, L., González, M., Land, M., Lao, V., Larimer, F., LiPuma, J.J., Mahenthiralingam, E., Malfatti, S.A., Marx, C.J., Parnell, J.J., Ramette, A., Richardson, P., Seeger, M., Smith, D., Spilker, T., Sul, W.J., Tsoi, T.V., Ulrich, L.E., Zhulin, I.B., Tiedje, J.M.: *Burkholderia xenovorans* LB400 harbors a multi-replicon, 9.73-mbp genome shaped for versatility. Proc. Natl. Acad. Sci. USA **103** (2006) 15280–15287
9. Weka: Waikato environment for knowledge analysis. (`http://sourceforge.net/projects/weka/`)
10. Arredondo, T., Freund, W., Muñoz, C., Navarro, N., Quirós, F.: Fuzzy motivations for evolutionary behavior learning by a mobile robot. Lecture Notes in Artificial Intelligence **4031** (2006) 462–471